Lombok annotations and its usage

Description

Assuming that you know the significance of Lombok on how it reduces the boilerplate code, I have created this post just to have a quick glance on all the Lombok annotations and its usage. This will help in quickly referring the annotations and using it when required.

If you want a detailed explanation on any annotation, you can click on it and read the elaborated official documentation.

Annotation	Usage
@Getter	This annotation creates getter methods. It can either be used above the class or above any class field.
@Setter	This annotation creates setter methods. It can either be used above the class or above any class field.
@Data	This annotation is a combination of @ToString, @EqualsAndHashCode, @Getter, @Setter and @RequiredArgsConstructor
@EqualsAndHashCode	This annotation creates both equals() and hashCode() methods
@ToString	This annotation creates a toString() method. If you want to exclude any field then you can add exclude key. E.g @ToString(callSuper=true,exclude=”accountNo”)
@NonNull	This annotation will not accept null values. It can be applied to class field, method parameter and constructor parameter.
@RequiredArgsConstructor	Generates a constructor with required arguments. Required arguments are final fields and fields with constraints such as @NonNull.
@AllArgsConstructor	This annotation creates a constructor consisting of all fields.
@NoArgsConstructor	Using this annotation we can generate default constructor for our class.
@Synchronized	@Synchronized is a safer variant of the synchronized method modifier. Like synchronized, the annotation can be used on static and instance methods only. It operates similarly to the synchronized keyword, but it locks on different objects. The keyword locks on this, but the annotation locks on a field named $lock, which is private.
@Builder	If you are familiar with builder pattern, this annotation is used to generate a builder for all the fields of the annotated class

Annotation	Usage
@SneakyThrows	@SneakyThrows can be used to sneakily throw checked exceptions without actually declaring this in your method's throws clause. This somewhat contentious ability should be used carefully, of course. The code generated by lombok will not ignore, wrap, replace, or otherwise modify the thrown checked exception; it simply fakes out the compiler

Category

1. Lombok

Date Created
November 7, 2021
Author
kk-ravi144gmail-com